International Journal of Advances in Intelligent Informatics
Vol. 4, No 1, March 2018, pp. 11-20
ISSN 2442-6571

A novel intelligent approach for detecting DoS flooding attacks in software-defined networks

Majd Latah [a,1,*], Levent Toker [b,2]

[a] Department of Computer Science, Özyegin University, Çekmekoy, Istanbul, 34794, Turkey
[b] Department of Computer Engineering, Ege University, Bornova, Izmir, 35100, Turkey
[1] majd.latah@ozu.edu.tr; [2] levent.toker@ege.edu.tr
* corresponding author

ARTICLE INFO

Article history
Received December 27, 2017
Revised February 9, 2018
Accepted March 8, 2018

Keywords
Denial of Service (DoS)
Flooding Attacks
Software-Defined Networking (SDN)
Support Vector Machines (SVM)
Network Security

ABSTRACT

Software-Defined Networking (SDN) is an emerging networking paradigm that provides an advanced programming capability and moves the control functionality to a centralized controller. This paper proposes a two-stage novel intelligent approach that takes advantage of the SDN approach to detect Denial of Service (DoS) flooding attacks based on calculation of packet rate as the first step and followed by Support Vector Machine (SVM) classification as the second step. Flow concept is an essential idea in OpenFlow protocol, which represents a common interface between an SDN switch and an SDN controller. Therefore, our system calculates the packet rate of each flow based on flow statistics obtained by SDN controller. Once the packet rate exceeds a predefined threshold, the system will activate the packet inspection unit, which, in turn, will use the (SVM) algorithm to classify the previously collected packets. The experimental results showed that our system was able to detect DoS flooding attacks with 96.25% accuracy and 0.26% false alarm rate.

1. Introduction

Software-defined networking (SDN) is a novel networking architecture that brings advanced programming capabilities leading to more enhanced behavior achieved by a controlling plane. Unlike traditional network architectures, SDN architecture manages forwarding elements through a centralized controller that maintains a global view of the network. Consequently, this approach opens a door for developing more flexible network services and applications. On the other hand, the increased usage of the Internet, requires us to protect all network devices involved in the communication process. Denial of Service (DoS) attack is a major threat in which the intruder attempts to prevent legitimate users from accessing network resources. This study uses an artificial intelligent based approach to deal with DoS attacks by taking the advantage of the central management provided by the SDN approach.

SDN approach is the fruit of many years' effort to create a programmable network managed by a centralized controller. Separation between control plane and data plane requires an interface between the controller and forwarding elements. OpenFlow represents a vendor-independent interface which translates the high-level orders sent by the controllers to low-level behaviors that can be understood by the switches, that handles the L2-L4 network flows. However, OpenFlow has to be extended in order to handle the L5-L7 flows [1]. The control plane provides a global view of the network that provides help in achieving more enhanced control mechanism for the forwarding plane. A basic SDN architecture is illustrated in Fig. 1.

In SDN architecture, the forwarding decision is made based on flow entries. Each flow entry consists of: (i) the header fields, used for matching an incoming packet with this flow, (ii) counters, provide useful statics related to that flow and (iii) actions that will be applied to the matched packets [2]. The controller is responsible for adding, modifying and deleting the flow entries. Furthermore, the controller can make proactive or reactive decisions [3]. Proactive decisions can be used in multipath forwarding applications which require less switch-controller communication where all flow entries are already added before the incoming traffic arrives at the switch. While in reactive decisions, unmatched packets are encapsulated in *PACKET_IN* messages and forwarded to the controller which, in turn, will reply to these messages with the appropriate decision by adding a new flow entry to the related switches in order to handle this type of packets by the switch itself [3]. Security applications are considered an example of the reactive approach.

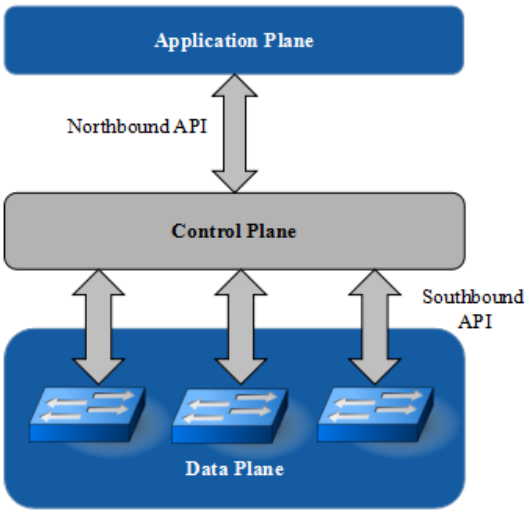

Fig. 1. A basic SDN architecture

In addition, SDN architecture introduces new security challenges (which are out from the scope of this work) such as targeting the controller by programming vulnerabilities, error configurations and DoS attacks on the controller-switch secure channel [4]. Due to the fact that the data plane is managed remotely, both DoS attack and its distributed version (DDoS) have a significant impact in SDN paradigm. As a result, it will flood the switch with illegitimate flow entries and consequently, will prevent the controller from responding to legitimate flows [5].

A simple way to detect DoS attack in SDN is to monitor the volume of each flow using flow-based statistics provided by the SDN controller. Such a mechanism was introduced in YuHunag et al. [6] where the detection loop consists of two stages. If the total number of the received packets exceeds 3000 packets per second, the system will move onto the next stage which the incoming packets will be dropped when the total number of the received is equal or higher than 800 packets per second for 5 times continuously. Braga et al. [7] employed Self-Organizing Maps (SOM) for detecting the distributed version of DOS attack. The following features were obtained from each collected flow: Average of Packets per flow (APf), Average of Bytes per flow (ABf), Average of Duration per flow (ADf), Percentage of Pair-flows (PPf), Growth of Single-flows (GSf), and Growth of Different Ports (GDP). This approach showed a lower overhead compared to other traditional approaches.

Chen and Yu [8] used a back-propagation neural network in SDN paradigm to detect DoS attacks. The experimental study showed that the growing of network topology will lead to an increase in the attack detection rate. Wang et al. [9] proposed a graphical probabilistic inference model with an updating phase in order to cope with dataset shift problem which appears when the network traffic conditions differ from the actual conditions.

Taekyoung and Yanghee [10] employed a content-oriented networking architecture in which each access router is called an “agent”. Each host belongs to a specific agent and sends a content request to

that agent, which, in turn, will deliver the requested contents. A solicitation of the contents is made by the agent in order to prevent arbitrary packet forwarding, which improves the security of the network. Furthermore, the agent can keep tracking the requested contents as well as the host that creates these requests. The agent can discover DDoS attacks if the rate of content requests exceeds a threshold or a malicious request is discovered.

Seungwon et al. [11] proposed a simple algorithm inspired by SYN cookie algorithm for handling TCP-ACK packets. This type of packets is used in TCP handshake and can be used to perform a DDoS attack on a targeted server. Moreover, fuzzy logic through its ability to derive conclusions from insufficient or imprecise data performs an effective way to deal with DoS attacks [12]–[14]. On the other hand, information theory-based metrics play a significant role in the detection of DoS attacks due to their low computation overhead [15]. During a DoS attack, entropy values decrease significantly [16].

A fast entropy algorithm was introduced by David and Thomas [17] in order to implement a lightweight DDoS detection method. The first step in this algorithm is the flow aggregation, in which the flow count of each connection is collected over a particular time interval. The next step is entropy calculation. The last step is applying an adaptive algorithm to improve the detection accuracy. However, the main disadvantage of this approach is that it mainly depends on calculation the entropy values through a specific time-window, which forms a potential overhead in the case of being adopted by an SDN controller.

In this study, we will employ the flow statistics, which can be easily obtained by the SDN controller, in order to calculate the packet rate of each flow and subsequently discriminate the potential malicious flows that may contribute to a DoS flooding attack. After that, we use Support Vector Machines (SVM) algorithm for classifying the collected traffic.

2. Method

2.1. Support Vector Machine (SVM)

Support vector machine (SVM) [18] is a statistical machine learning algorithm that uses the idea of finding the optimal hyperplane separation of labeled instances in a given data set which leads to a better generalization of the unseen data (Fig. 2). SVM and its various modified versions [19], [20] are widely used in intrusion detection systems.

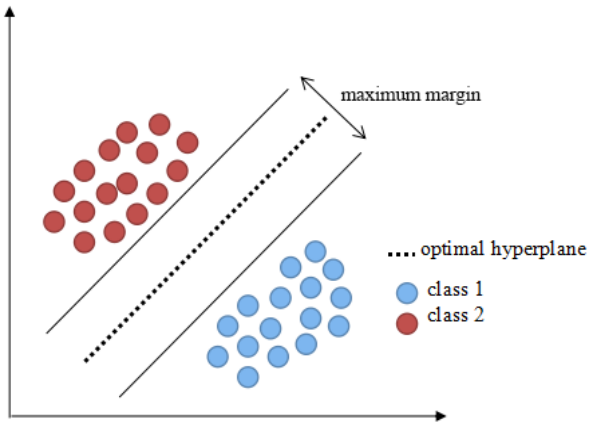

Fig. 2. Linear support vector machine.

For a given data set $Z\{(x_1, y_1),(x_2, y_2),...,(x_n, y_n)\}$ where $x_i \in R^n$ and $y_i \in \pm 1$ which is the label of the instances and $i = 1, 2, ..., n$ the set of hyperplanes in SVM can be written as:

$$w \cdot x_i + b \geq +1 ; y_i = +1 \quad (1)$$

$$w \cdot x_i + b \leq -1 ; y_i = -1 \quad (2)$$

These hyperplanes can be combined by the following inequality:

$$y_i (w \cdot x_i + b) \geq 1 ; \ 1 \leq i \leq n \quad (3)$$

This optimization problem can be written as:

$$\underset{w,b}{Min}: \frac{1}{2} w^T w \quad subject to \ y_i (w \cdot x + b) \geq 1 \quad (4)$$

A non-separable case can be handled by introducing slack variables $\xi_i \geq 0, \ i = 1, \ldots, n$ which allows us to select a hyperplane with minimum errors:

$$\underset{w,b,\xi}{Min}: \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i \quad subject to \ y_i (w \cdot x + b) + \xi_i \geq 1 \ ; \ \xi_i \geq 0 \quad (5)$$

where C is a regularization parameter which affects the trade-off between the allowed errors in training phase and margin size. Large values of C lead to a smaller-margin which may cause an over-fitting case. Furthermore, to deal with nonlinear cases, a kernel function can be used to map the data set to a higher-dimensional space as shown as (6).

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (6)$$

The most commonly used kernel functions are polynomial and Radial basis function (RBF). The RBF kernel which is used by our SVM classifier is given by (7).

$$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2) \quad , \quad \gamma > 0 \quad (7)$$

where γ is the kernel parameter.

2.2. Dataset

This experimental study is conducted by generating both volume-based and protocol-based DoS attacks and taking into consideration the case in which the attacker may send benign traffic besides the malicious one. By this means, a data set which consists of 321 instances of normal traffic and 639 instances of attack traffic, was generated during the evaluation stage where D-ITG tool [21] is used to generate UDP, ICMP flooding attacks whereas *scapy* library (secdev.org) is used for generating SYN flooding attacks. In this study, we generate DoS attacks with different rates (100,500,1000) pps. Using D-ITG tool one can generate UDP, ICMP floods with different payload size.

As shown in Fig. 3, we generated a UDP flood to a destination host which has 10.0.0.4 IP address with constant payload size (500 bytes) and constant packet rate (100 pps) for 1 second (1000 ms). We also assume that each IP address of each packet was not spoofed.

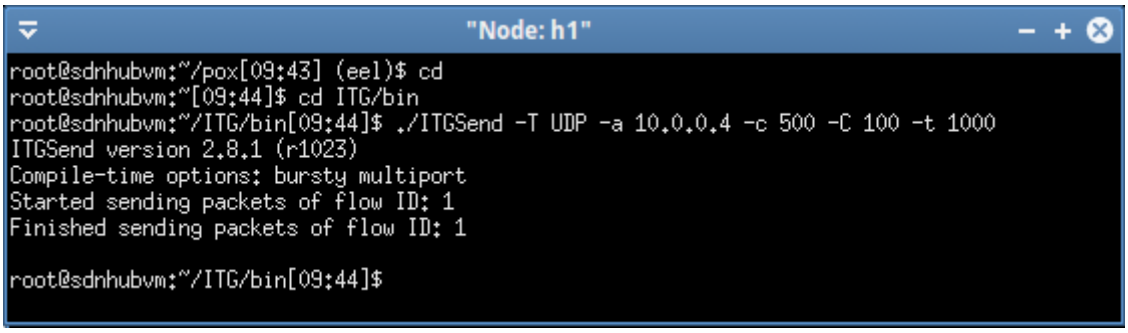

Fig. 3. Sending a UDP flood using D-ITG tool.

Similarly, *scapy* library is used to generate SYN floods. As shown in Fig. 4, we lunch a SYN flood attack via Python/*scapy* script. Finally, we wrote a Python script that collects packet features from each corresponding host for periodically every 4-seconds based on *scapy* library which actually comes with a built-in packet sniffer and used widely as packet manipulation, capture, modification and replay library [22].

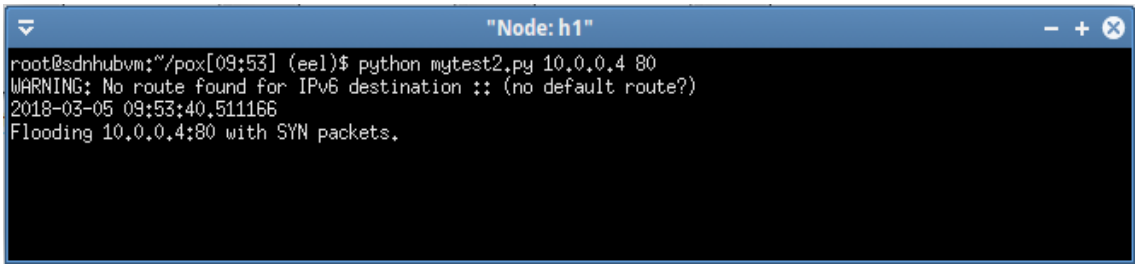

Fig. 4. Sending a SYN flood using python script with *scapy* library.

In addition, the collected features were normalized using equation (8).

$$x_{new} = \frac{x_{old} - x_{\min}}{x_{\max} - x_{\min}} \tag{8}$$

2.3. Experimental Setup

In order to evaluate the system, we used Mininet, an open-source SDN emulator where the experiment was conducted in a virtual environment (Linux Ubuntu 14.04/Virtual Box) on Intel i5 machine with 12 GB of RAM. The controller part was implemented using POX controller. The packet inspection unit, on the other hand, was implemented using a combination of *scapy* and *sklearn* Python libraries. As shown in Fig. 5, our testbed consists of 9 virtual hosts connected to different OpenFlow switches. The bandwidth of each link is 1 Mb/s. The connection between the SDN controller and the hosts was done locally where the SDN controller can communicate with any device and trigger the packet inspection unit. By this means, one can use the "util/m" script provided by Mininet to control Mininet hosts. For instance, we can write the following command in Linux Ubuntu terminal to access host1 in Mininet environment:

```
-> mininet/util/m h1 ifconfig
```

Instead of running *ifconfig* command, the SDN controller (i.e. POX controller) runs a Python script, which represents our packet inspection unit. In this context, it is worth noting that securing the communication between the SDN controller and the corresponding host is out of the scope of this study.

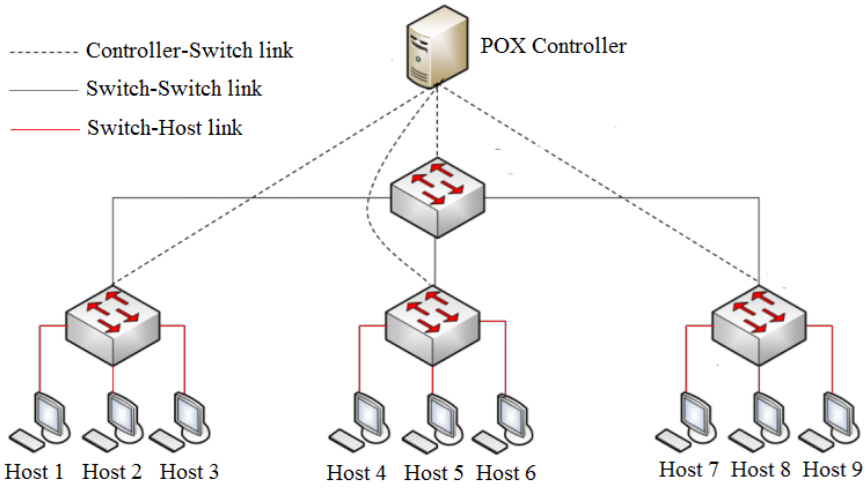

Fig. 5. Testbed used for our experiment.

2.4. Performance Evaluation

To evaluate our classifier, we applied 10-fold cross-validation where the dataset is partitioned into ten equally-sized folds. We train the model on 90% of the samples and then test the class labels of the remaining 10%. This procedure is repeated 10 times. The accuracy for each fold is determined by (9).

$$Accuracy = \frac{TP+TN}{(TP+TN+FN+FP)} \tag{9}$$

where True Positives (TP) is the total number of abnormal traffic instances correctly classified as abnormal traffic; True Negatives (TN) is the total number of normal traffic instances correctly classified as normal traffic; False Positives (FP) is the total number of normal traffic instances falsely classified as abnormal traffic and False Negatives (FN) is total number of abnormal traffic instances falsely classified as normal traffic. The accuracy of the whole folds is used to compute the final accuracy. In addition, the false alarm rate for each fold is calculated by (10).

$$False\ alarm\ rate = \frac{FP}{TN+FP} \tag{10}$$

3. The Proposed Method

In this section, we present our proposed approach which consists of two stages.

3.1. The first stage

In this stage includes packet rate calculation based on the flow statistics collected by the SDN controller. Once the packet rate exceeds a predefined threshold, the controller will activate the packet inspection unit which is responsible for collecting packet statistics, applying SVM algorithm and notifying the SDN controller about the classification results. As depicted in Fig. 6, the packet inspection unit is implemented on each host separately, where the controller can activate the packet inspection unit remotely and take the proper decision based on a binary classification step conducted by a previously trained SVM classifier.

In volume-based attacks, the intruder sends numerous packets from different sources to the target server. This consumes a huge amount of network bandwidth. While in protocol-based attacks, the attacker exploits a weakness in the Internet protocols which can lead to a complete denial of service in targeted server's resources. The packet rate is a significant metric which was used in [12] for implementing a host-based DoS attack detection system using Mamdani's fuzzy inference model. In our

proposed method, however, we calculate the packet rate using equation (11) based on the flow statistics collected periodically every 10 seconds by the SDN controller.

$$Packet\ rate = \frac{Total\ trasmitted\ packets\ by\ current\ flow}{Flow\ duration} \quad (11)$$

If the packet rate is greater than or equal to 100 packets per second, then the system will move into the next stage and it will be discussed in the next subsection. The reason of selecting low values of the packet rate (to trigger the next stage) is that during our experimental study we found that the performance of the selected controller (i.e. POX controller) was affected significantly by flooding attacks that exceeds 100 packets per second, and therefore we concluded that it can be more efficient to detect these type of DoS attacks starting from the mentioned threshold.

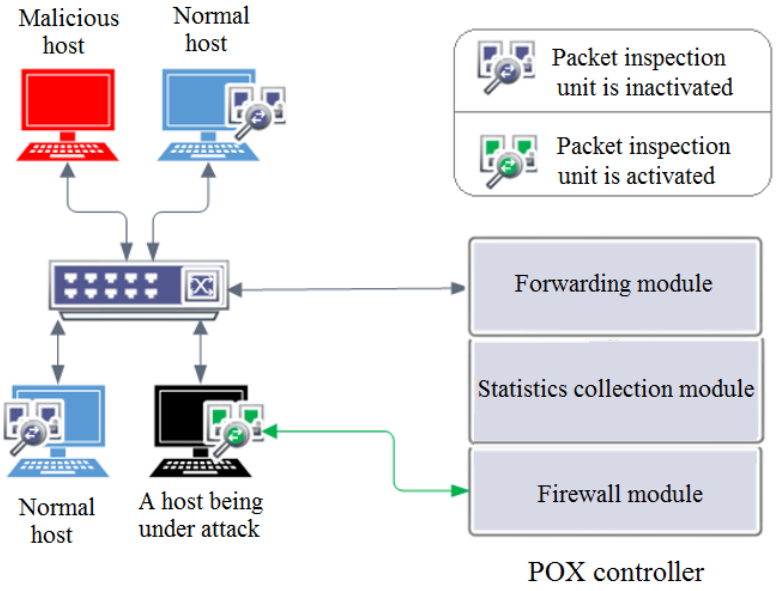

Fig. 6. An overview of our proposed approach.

3.2. The Second Stage

The aim of the second stage is to reduce the false alarm rate. Once the packet rate exceeds the predefined threshold, the controller will activate the packet inspection unit. At this point, packet-based statistics will be collected during a 4-second period by the packet inspection unit so that it can be used by our SVM classifier. As a consequence, it will determine whether the attack is happened or not. The collected features in the second stage are shown at Table 1.

Table 1. Collected features for the classification stage

Feature	Description
Total_pkt	Total number of collected packets
Total_payload	Total payload size
SYN_no	Total number of SYN flags
ACK_no	Total number of ACK flags
TCP_no	Total number of TCP packets
UDP_no	Total number of UDP packets
ICMP_no	Total number of ICMP packets
Protocol_no	Total number of used protocols.
Port_no	Total number of used ports.

4. Results and Discussion

Using our SVM classifier, the accuracy and false alarm rate for each fold is shown at Table 2. The average accuracy for 10-fold cross validation is 96.25% whereas the average false alarm rate is 0.26%. As a result, the SVM classifier showed very low false positives compared to the total number of false negatives.

Table 2. Results of 10-fold cross-validation

	1-Fold	2-Fold	3-Fold	4-Fold	5-Fold	6-Fold	7-Fold	8-Fold	9-Fold	10-Fold
TP	57	70	71	69	63	50	57	66	76	59
TN	37	26	25	27	32	42	38	3	19	37
FN	1	0	0	0	1	4	1	27	1	0
FP	1	0	0	0	0	0	0	0	0	0
Accuracy (%)	97.91	**100**	**100**	**100**	98.96	95.83	98.96	71.88	98.96	**100**
False alarm rate (%)	2.63	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Fig. 7 shows the detection results based on the implementation of our two-stage approach where the SDN controller updates the flow table of the corresponding switch by sending a flow mod instruction that deletes the malicious flows. Thereafter, the controller updates the firewall module by adding the corresponding malicious source MAC address in the blocked list.

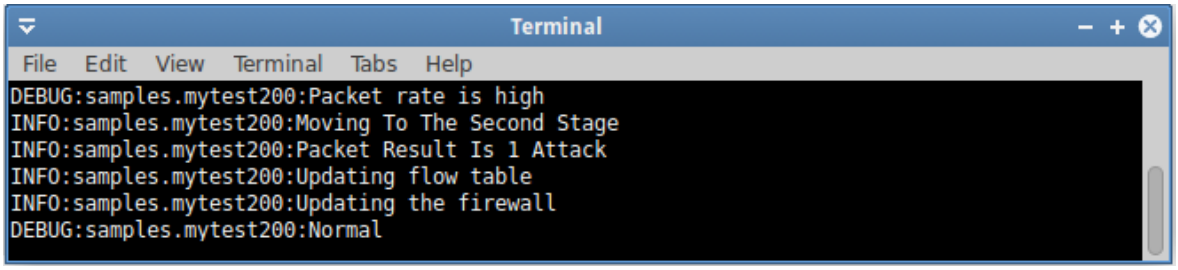

Fig. 7. Results for implementing our proposed system based on the POX controller.

In addition, Table 3 shows a comparison between the SVM approach and the other supervised machine learning approaches. In terms of false alarm rate, the SVM approach has shown the best results. In terms of accuracy, the random forest classifier has shown a better accuracy compared to SVM approach. However, the highest accuracy was achieved using a Multilayer Perceptron (MLP) neural network (a single hidden layer with 4 neurons).

Table 3. Comparision between SVM and other supervised machine learning approaches

Classifier	10-fold cross-validation	
	Accuracy	*False alarm rate*
Logistic regression	90.417%	5.24%
kNN (k-nearest neighbors)	94.167%	8.87 %
Naïve bayes	94.375%	5.52 %
Decision trees	95.354%	6.55 %
SVM (C=1, γ = 1000)	96.25%	**0.26 %**
Random forest	97.812%	3.93 %
Neural network (MLP)	**98.958%**	1.004%

5. Conclusion

By this study we have proposed, designed and implemented a two-stage novel DoS flooding attack detection method in SDN paradigm. In the first stage, when the packet rate exceeds a predefined threshold, the system will move onto the next stage which employs SVM algorithm to investigate the suspected traffic. In our experimental studies, we showed that the controller can communicate with any

device and trigger the next stage (packet inspection unit) in Mininet environment. Our proposed system was able to detect DoS flooding attacks with 96.25% accuracy and 0.26% false alarm rate based on SVM classifier with RBF-kernel. Compared to other classifiers, the SVM approach has shown the lowest false positive rate. In addition, the MLP-neural network approach has shown the highest accuracy. We conclude that the integration between the SDN approach and machine learning techniques formed a promising solution for providing more secure networks. Our future work will be focused on using the SDN paradigm for detecting application-layer DoS attacks as well as investigating the efficiency of other multi-stage approaches in order to identify more complex types of network attacks.

Acknowledgment

We would like to thank the Turkish government scholarship for international students for supporting the corresponding author.

References

[1] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "A virtual SDN-enabled LTE EPC architecture: A case study for S-/P-gateways functions," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, 2013, pp. 1–7, doi: https://doi.org/10.1109/SDN4FNS.2013.6702532.

[2] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Comput. Networks*, vol. 72, pp. 74–98, 2014, doi: https://doi.org/10.1016/j.comnet.2014.07.004.

[3] P. Goransson, C. Black, and T. Culver, *Software defined networks: a comprehensive approach*, 2nd Editio. Morgan Kaufmann, 2016, available at: https://www.elsevier.com/books/software-defined-networks/goransson/978-0-12-804555-8.

[4] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, "Securing software defined networks: taxonomy, requirements, and open issues," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 36–44, 2015, doi: https://doi.org/10.1109/MCOM.2015.7081073.

[5] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Comput. Secur.*, vol. 53, pp. 79–108, 2015, doi: https://doi.org/10.1016/j.cose.2015.05.006.

[6] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, "A novel design for future on-demand service and security," in *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, 2010, pp. 385–388, doi: https://doi.org/10.1109/ICCT.2010.5689156.

[7] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, 2010, pp. 408–415, doi: https://doi.org/10.1109/LCN.2010.5735752.

[8] X.-F. Chen and S.-Z. Yu, "CIPA: A collaborative intrusion prevention architecture for programmable network and SDN," *Comput. Secur.*, vol. 58, pp. 1–19, 2016, doi: https://doi.org/10.1016/j.cose.2015.11.008.

[9] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software-defined networking," *Comput. Networks*, vol. 81, pp. 308–319, 2015, doi: https://doi.org/10.1016/j.comnet.2015.02.026.

[10] J. Suh, H. Choi, W. Yoon, T. You, T. T. Kwon, and Y. Choi, "Implementation of Content-Oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure," in *1st European NetFPGA Developers Workshop*, 2010, pp. 1–5, available at : https://mmlab.snu.ac.kr/publications/docs/2010_EU_netfpga_workshop_jhsuh.pdf.

[11] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 413–424, doi: https://doi.org/10.1145/2508859.2516684.

[12] N. Naik and P. Jenkins, "Fuzzy reasoning based windows firewall for preventing denial of service attack," in *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on*, 2016, pp. 759–766, doi: https://doi.org/10.1109/FUZZ-IEEE.2016.7737764.

[13] N. Naik, "Fuzzy inference based intrusion detection system: FI-Snort," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, 2015, pp. 2062–2067, doi: https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.306.

[14] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, 2014, pp. 167–171, doi: https://doi.org/10.1109/ICACT.2014.6778942.

[15] I. Özçelik and R. R. Brooks, "Deceiving entropy based DoS detection," *Comput. Secur.*, vol. 48, pp. 234–245, 2015, doi: https://doi.org/10.1016/j.cose.2014.10.013.

[16] K. Johnson Singh, K. Thongam, and T. De, "Entropy-Based Application Layer DDoS Attack Detection Using Artificial Neural Networks," *Entropy*, vol. 18, no. 10, p. 350, 2016, doi: https://doi.org/10.3390/e18100350.

[17] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow-based network traffic," *Procedia Comput. Sci.*, vol. 50, pp. 30–36, 2015, doi: https://doi.org/10.1016/j.procs.2015.04.007.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995, doi: https://doi.org/10.1007/BF00994018.

[19] Z. Zhang and H. Shen, "Application of online-training SVMs for real-time intrusion detection with different considerations," *Comput. Commun.*, vol. 28, no. 12, pp. 1428–1442, 2005, doi: https://doi.org/10.1016/j.comcom.2005.01.014.

[20] J. Yu, H. Lee, M.-S. Kim, and D. Park, "Traffic flooding attack detection with SNMP MIB using SVM," *Comput. Commun.*, vol. 31, no. 17, pp. 4212–4219, 2008, doi: https://doi.org/10.1016/j.comcom.2008.09.018.

[21] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG distributed internet traffic generator," in *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*, 2004, pp. 316–317, doi: https://doi.org/10.1109/QEST.2004.1348045.

[22] I. Mukhopadhyay, K. S. Gupta, D. Sen, and P. Gupta, "Heuristic Intrusion Detection and Prevention System," in *Computing and Communication (IEMCON), 2015 International Conference and Workshop on*, 2015, pp. 1–7, doi: https://doi.org/10.1109/IEMCON.2015.7344479.